Chapter 9. Configuring Your System

Contents

Notes:

Introduction

Each time DOS is started, it searches the root directory of the drive (from which it was started) for a special configuration file named CONFIG.SYS. If found, it reads the file and interprets the text commands within it.

Configuration Commands

The following commands can be included in the configuration file. If you add or change any of the configuration file commands, they will become effective the *next* time DOS is started.

BREAK Command

BREAK=ON/OFF

This command should only be used once in the configuration file. The default value is OFF, and causes DOS to check for Ctrl-Break being entered at the keyboard only when DOS is performing screen, keyboard, printer or Asynchronous Communication Adapter operations. With this setting, it may not be possible to cancel an executing program by using Ctrl-Break unless the program causes DOS to perform one of those four operations. Specifying ON causes DOS to check for Ctrl-Break whenever it performs *any* function for a program. This allows you to "break" out of programs that perform few (or no) screen, keyboard, printer, or auxiliary device operations (such as compilers). The ON/OFF state set in the configuration file can later be changed by issuing a BREAK command (see Chapter 6).

BUFFERS Command

BUFFERS=*xx*

Where *xx* is a number between 1 and 99. This is the number of disk buffers that DOS should allocate in memory when it starts up. The default value is 2, and this value will remain in effect until DOS is restarted with a different value specified in the configuration file.

What Is a Buffer

A disk buffer is a block of memory that DOS uses to hold data being read from, or written to a disk (fixed disk or diskette), when the amount of data being transferred is not an exact multiple of the sector size. For example, if an application reads a 128-byte record from a file, DOS will read the entire sector into one of its buffers, locate the correct 128-byte record in the buffer, and move the record from the buffer into the application's area of memory. It then marks that buffer as having been used recently. On the next request to transfer data, DOS will attempt to use a different buffer. In this way, all of the buffers will eventually contain the most recently-used data. The more buffers DOS has, the more data will be in memory.

Read/Write Requests

Each time DOS is requested to read or write a record that is not an exact multiple of the sector size, it first looks to see if the sector containing that record is already in a buffer. If not, it must read the sector as described above. But if the data is already in a buffer, then DOS can simply transfer the record to the application's area without the need to read the sector from the disk—this saves time. This savings is realized on both reading and writing records, since DOS must first read a sector before it can insert a record your application is attempting to write.

Random/Sequential Applications

For applications that read and write records in a random fashion (such as many Basic and data base applications), the likelihood of finding the correct record already in a buffer increases if DOS has more buffers to work with. This can greatly speed up the performance of those applications.

For applications doing sequential reads and writes, however (read an entire file, write an entire file), there is little advantage to having a large number of buffers allocated.

Because all applications are different, there is no specific number of buffers that will serve all applications equally well. If your applications do little random reading and writing of records, the system default of 2 buffers (if you do not specify BUFFERS= in your configuration file) should be sufficient.

However, if you use data-base type applications, or run programs that perform a lot of random reads and writes of records, you will want to increase the number of DOS buffers. The "best" number of buffers for your particular application can only be determined by using different values until the best performance is achieved. For most data base applications, a value between 10 and 20 buffers will usually provide the best results.

Beyond that point, the system may appear to start running slower - this is because, with a very large number of buffers it can take DOS longer to search all the buffers for the record than it would take to read the record from disk.

Size of Your Computer

The final consideration in determining the number of buffers to allocate is the memory size of your computer. Since each additional buffer increases the resident size of DOS by 528 bytes, the amount of memory available to the application is reduced by that amount. Therefore, additional buffers may actually cause some applications to slow down, since there is less memory in which the application itself can keep data—this could result in more frequent reads and writes than would otherwise be necessary.

In summary, the optimum number of buffers must be determined by *you*, based on:

1. The types of applications most often used
2. The memory size of your computer
3. Your analysis of system performance when using your applications with different numbers of buffers allocated.
4. For computers with fixed disks, we recommend a minimum of BUFFERS=3.

DEVICE Command

DEVICE=[*d*:][*path*]*filename*[.*ext*]

This command allows you to specify the name of a file containing a device driver. During startup, DOS loads the file into memory as an extension of itself, and gives it control as described in "Installable Device Drivers" in Chapter 14. Please refer to that section for technical information about installable device drivers.

Loading Standard Device Drivers

The standard device drivers loaded by DOS support the standard screen, keyboard, printer, auxiliary device, diskette, and fixed disk devices. A clock driver is also loaded (see Chapter 14). You don't need to specify any DEVICE= commands for DOS to support these devices.

Replacing Standard Device Drivers

If you wish to use the "Extended Screen and Keyboard Control" features described in Chapter 13, you should create the file CONFIG.SYS on the disk you will be starting DOS from. The file should contain the command DEVICE=ANSI.SYS. This command causes DOS to replace the standard screen and keyboard support with the extended functions.

Installing Your Own Device Driver

For systems programmers and application developers—if you have written device drivers that you want DOS to load when it starts, include a DEVICE= command in the CONFIG.SYS file for each driver to be loaded.

FILES Command

FILES=*xx*

The maximum value for *xx* is 99.

Beginning with DOS Version 2.0, there is no need for an application to construct a special control block (FCB) in order to access a file. Instead, the program can simply specify an ASCII string consisting of drive specifier, complete directory path name and filename when opening or creating a file. DOS will locate the correct drive, directory and file, and will create and return a *handle*—merely a 16-bit binary value.

Accessing a File

All file accesses (reads, writes, close) can then be performed by telling DOS which handle to use. When an application opens a file in this manner, DOS constructs a control block in its own memory on behalf of the application, in an area that was set aside when DOS started. The size of this area (and consequently, the maximum number of files that can be concurrently open), depends on the value specified in the FILES= command.

The default value is FILES=8; that is, no more than 8 files can be open at the same time. There is no effect on the number of files that can be concurrently open using the traditional (OPEN FCB) functions. This default value is sufficient for the majority of operating environments. However, if applications are installed that result in error messages indicating an insufficient number of handles, the FILES= command should be used to provide DOS with additional handles.

Number of Files Opened

The value specified in FILES= becomes the new maximum number of files that DOS allows to be concurrently open.

If you specify FILES= in your configuration file, the size of the resident portion of DOS increases by 39 bytes for each additional file above the default value of 8. Consequently, the memory available to the application is reduced by the same amount. See function calls hex 3C through hex 46 in Appendix D for descriptions of the new file-handling functions.

SHELL Command

SHELL=[*d*:][*path*]*filename*[.*ext*]

This command allows you to specify the name and location of a top-level command processor that DOS initialization will load in place of COMMAND.COM.

System programmers who develop their own top-level command processor should remember to include provisions for handling interrupts hex 22, hex 23 and hex 24, and for reading and executing commands. Because the internal commands, batch processor, and EXEC function call (program loader) reside in COMMAND.COM. These functions will not be available to the user unless they are duplicated in your command processor.